

10076667, 031402

1 CLAIMS

2

3 1. A method comprising:

4 receiving an indication of an application programming interface definition

5 that is written in a markup language; and

6 transforming the interface definition into a non-markup language source

7 file.

8

9 2. A method as recited in claim 1, wherein the source file comprises a

10 component object model (COM) header file.

11

12 3. A method as recited in claim 1, wherein the source file comprises

13 code mapping a set of enumeration values to corresponding string values.

14

15 4. A method as recited in claim 1, further comprising transforming the

16 application programming interface definition into a documentation file that

17 describes the contents of the non-markup language source file.

18

19 5. A method as recited in claim 1, wherein the source file comprises a

20 proxy object code file.

21

22

23

24

25

1 6. A method as recited in claim 1, further comprising transforming the
2 application programming interface definition into a test proxy object code file,
3 wherein the test proxy object code file includes a plurality of test proxies to assist
4 in testing the non-markup language source file.

5
6 7. A method as recited in claim 1, wherein receiving the indication
7 comprises receiving a filename of the application programming interface
8 definition.

9
10 8. A method as recited in claim 1, wherein receiving the indication
11 comprises receiving the application programming interface definition.

12
13 9. A method comprising:
14 receiving an indication of an interface definition, wherein the interface
15 definition includes a plurality of constructs;
16 transforming the interface definition into data for a first file; and
17 transforming the interface definition into data for a second file, wherein the
18 data for the first file is different than the data for the second file.

19
20 10. A method as recited in claim 9, wherein the first file and the second
21 file are each different ones of the following types of files: a component object
22 model (COM) header file, a mapping file to map enumeration values to
23 corresponding string values, a proxy object code file, and a documentation file.

1 **11.** A method as recited in claim 9, further comprising:
2 determining that the interface definition has been changed;
3 re-transforming the interface definition into data for a third file, wherein the
4 third file is the same type of file as the first file; and
5 re-transforming the interface definition into data for a fourth file, wherein
6 the fourth file is the same type of file as the second file.

7
8 **12.** A method as recited in claim 11, wherein determining that the
9 interface definition has been changed comprises automatically detecting that the
10 interface definition has been changed.

11
12 **13.** One or more computer readable media having stored thereon a
13 plurality of instructions that, when executed by a transformation engine, causes the
14 transformation engine to:

15 access a plurality of constructs in an application programming interface
16 description, wherein the description is written in an extensible markup language
17 (XML) format; and

18 transform each of the plurality of constructs into code for a component
19 object module (COM) application programming interface header file.

20
21 **14.** One or more computer readable media as recited in claim 13,
22 wherein the transformation engine comprises a series of instructions executed by
23 one or more processors.

1 **15.** One or more computer readable media as recited in claim 13,
2 wherein the plurality of instructions include instructions to:

3 check whether a declare enumeration construct is to be transformed into a
4 series of manifest constants or into a component object model enumeration
5 declaration; and

6 transform the enumeration construct into either the series of manifest
7 constants or the component object model enumeration declaration based on the
8 checking.

9
10 **16.** One or more computer readable media as recited in claim 13,
11 wherein the plurality of instructions include instructions to transform a declare
12 enumeration construct into a series of manifest constants.

13
14 **17.** One or more computer readable media as recited in claim 13,
15 wherein the plurality of instructions include instructions to transform a declare
16 enumeration construct into a component object model enumeration declaration.

17
18 **18.** One or more computer readable media as recited in claim 13,
19 wherein the plurality of instructions include instructions to transform a declare
20 function construct into a component object model function declaration.

21
22 **19.** One or more computer readable media as recited in claim 13,
23 wherein the plurality of instructions include instructions to transform a declare
24 class object construct into a component object model class object ID declaration.
25

20. One or more computer readable media as recited in claim 13,
wherein the plurality of instructions include instructions to transform a declare
interface construct into a component object model forward class declaration.

21. One or more computer readable media as recited in claim 13,
wherein the plurality of instructions include instructions to transform a declare
data structure construct into a component object model data structure declaration.

22. One or more computer readable media as recited in claim 13,
wherein the plurality of instructions include instructions to transform a declare
macro construct into a component object model manifest constant.

23. A method comprising:
receiving an indication of an application programming interface
description, wherein the description is written in an extensible markup language
(XML) format;
identifying a plurality of constructs in the application programming
interface description; and
transforming each of the plurality of constructs into code for a component
object module (COM) application programming interface header file.

24. A method as recited in claim 23, wherein the code for the
component object module comprises C source code.

1 **25.** A method as recited in claim 23, wherein the code for the
2 component object module comprises C++ source code.

3
4 **26.** A method as recited in claim 23, wherein the transforming
5 comprises transforming a declare enumeration construct into a series of manifest
6 constants.

7
8 **27.** A method as recited in claim 23, wherein the transforming
9 comprises transforming a declare enumeration construct into a component object
10 model enumeration declaration.

11
12 **28.** A method as recited in claim 23, wherein the transforming
13 comprises transforming a declare function construct into a component object
14 model function declaration.

15
16 **29.** A method as recited in claim 23, wherein the transforming
17 comprises transforming a declare class object construct into a component object
18 model class object ID declaration.

19
20 **30.** A method as recited in claim 23, wherein the transforming
21 comprises transforming a declare interface construct into a component object
22 model forward class declaration.

1 **31.** A method as recited in claim 23, wherein the transforming
2 comprises transforming a declare data structure construct into a component object
3 model data structure declaration.

4
5 **32.** A method as recited in claim 23, wherein the transforming
6 comprises transforming a declare macro construct into a component object model
7 manifest constant.

8
9 **33.** One or more computer readable media having stored thereon a
10 plurality of instructions that, when applied by a transformation engine, causes the
11 transformation engine to perform acts comprising:

12 receiving an indication of an application programming interface
13 description, wherein the description is written in an extensible markup language
14 (XML) format;

15 identifying one or more enumeration declaration constructs in the
16 application programming interface description; and

17 transforming each of the enumeration declaration constructs into a mapping
18 of enumeration values to corresponding string values.

19
20 **34.** One or more computer readable media as recited in claim 33,
21 wherein the transformation engine comprises a series of instructions executed by
22 one or more processors.
23
24
25

1 **35.** One or more computer readable media having stored thereon a
2 plurality of instructions that, when applied by a transformation engine, causes the
3 transformation engine to perform acts comprising:

4 receiving an indication of an application programming interface
5 description, wherein the description is written in an extensible markup language
6 (XML) format;

7 identifying a plurality of constructs in the application programming
8 interface description; and

9 transforming each of the plurality of constructs into code for a test proxy
10 file to be used in testing a component object module (COM) application
11 programming interface header file generated from the application programming
12 interface description.

13
14 **36.** One or more computer readable media as recited in claim 35,
15 wherein the transformation engine comprises a series of instructions executed by
16 one or more processors.

17
18 **37.** One or more computer readable media as recited in claim 35,
19 wherein each of the plurality of constructs comprises a declare interface construct.

20
21 **38.** A computer-readable medium having stored thereon a data structure
22 comprising:

23 an id attribute field that contains data identifying an application
24 programming interface description construct;

1 a plurality of construct fields that contain data describing the application
2 programming interface; and

3 a field functioning to identify the end of the data structure.

4
5 **39.** A computer-readable medium as recited in claim 38, wherein the
6 plurality of construct fields include one or more declare enumeration construct
7 fields.

8
9 **40.** A computer readable media as recited in claim 39, wherein each
10 declare enumeration construct field includes:

11 a plurality of declare enumeration member constructs; and

12 an enumeration flag attribute that identifies whether the plurality of declare
13 enumeration member constructs are to be transformed into a series of manifest
14 constants or transformed into a component object model enumeration declaration.

15
16 **41.** A computer-readable medium as recited in claim 38, wherein the
17 plurality of construct fields include one or more declare function construct fields.

18
19 **42.** A computer-readable medium as recited in claim 38, wherein the
20 plurality of construct fields include one or more declare class object construct
21 fields.

22
23 **43.** A computer-readable medium as recited in claim 38, wherein the
24 plurality of construct fields include one or more declare interface construct fields.

1 **44.** A computer readable media as recited in claim 43, wherein each
2 declare interface construct field includes:

3 one or more declare method constructs, wherein each declare method
4 construct stores data identifying a method corresponding to the interface defined
5 by the declare interface construct field; and

6 wherein each declare method construct includes one or more declare
7 parameter construct fields, wherein each declare parameter construct field stores
8 data identifying a parameter of the method.

9
10 **45.** A computer-readable medium as recited in claim 38, wherein the
11 plurality of construct fields include one or more declare data structure construct
12 fields.

13
14 **46.** A computer-readable medium as recited in claim 38, wherein the
15 plurality of construct fields include one or more declare macro construct fields.